BIGDATA TEAM

Machine Learning Course

MEGAFON

Time Series Analysis

Neychev Radoslav
ML Instructor, BigData Team
Research Scientist, MIPT

27.08.2018, Moscow, Russia

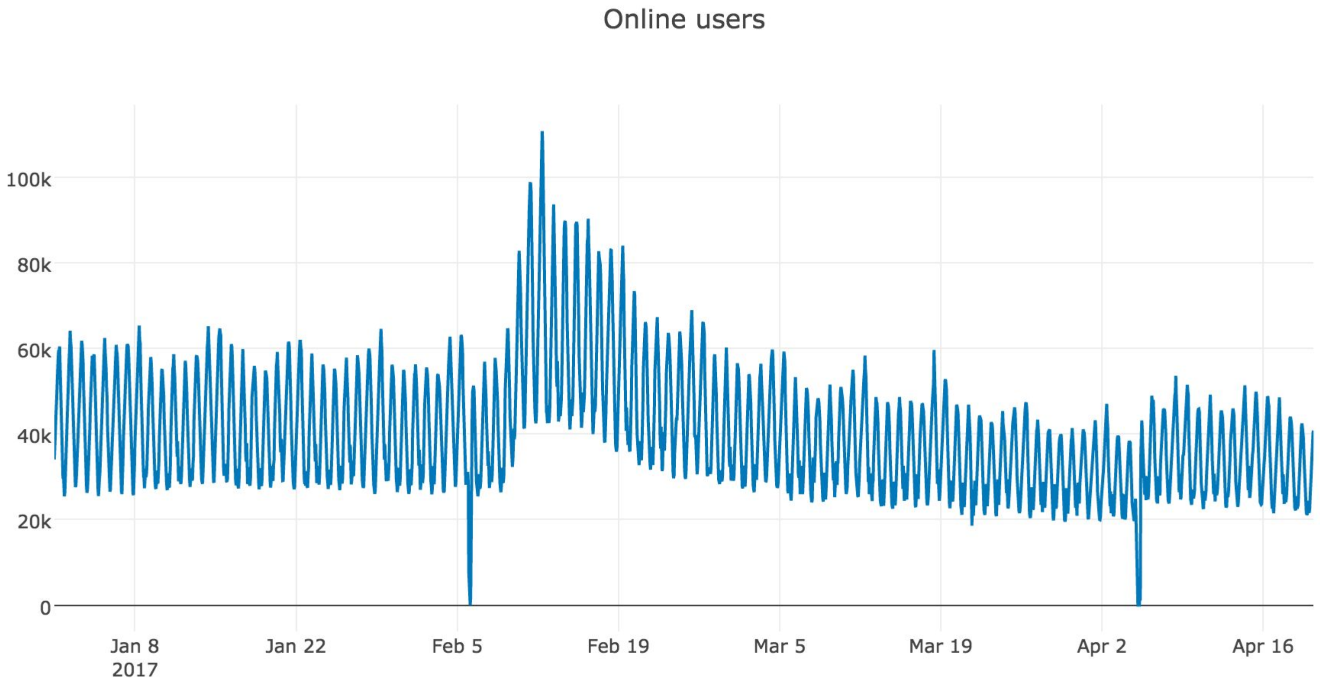
Outline

1. Time Series as data with time hierarchy.
2. Classical approaches: AR, MA, ARIMA, exp. smoothing etc.
3. Practice.
4. Feature engineering for Time Series analysis.
5. Modern approaches.
6. (extra) Neural networks (CNN & RNN) in TS analysis.
7. More practice.

Materials: http://bit.ly/ml4megafon_august18_public

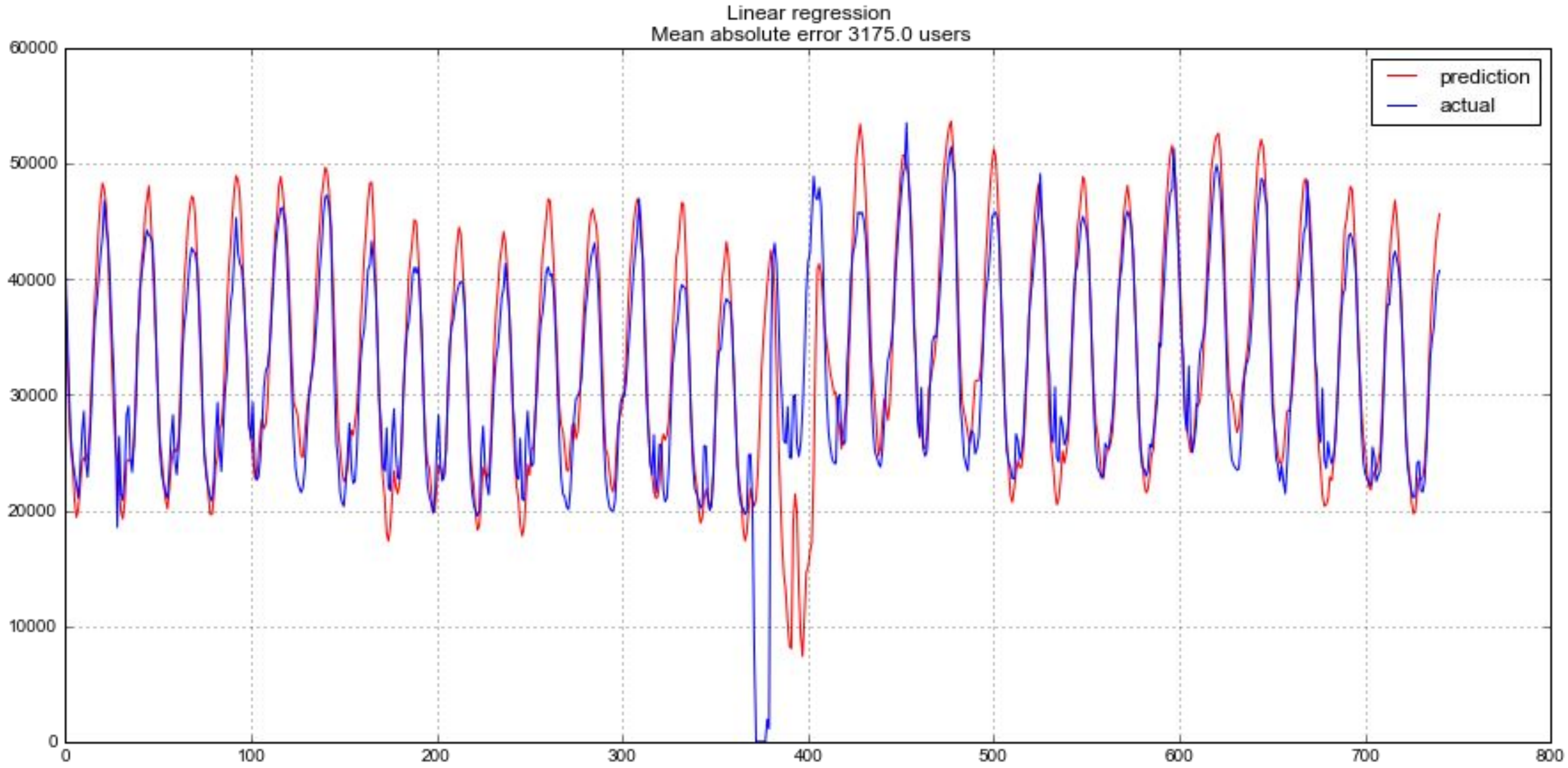
Credits: these slides are based on ODS open ML course materials and several blog posts.

Time Series -- series of data observations with time hierarchy.



Time Series: intro

How to measure the prediction quality?

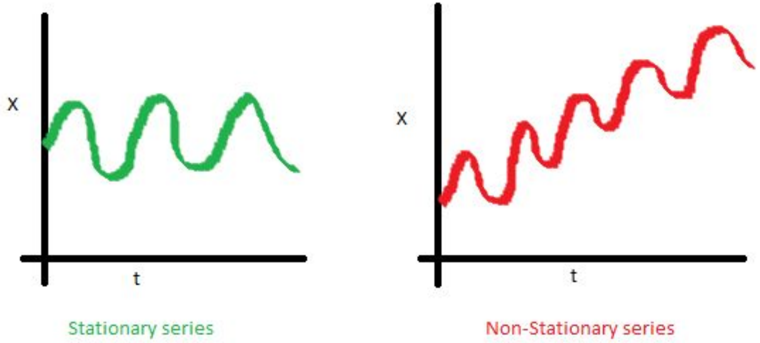


Time Series: metrics in forecasting

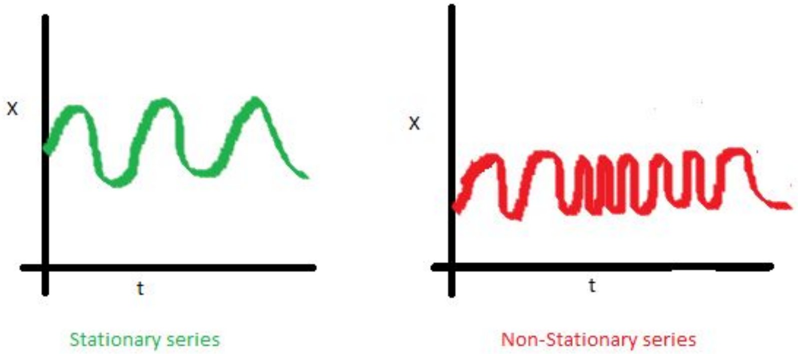
- R squared (coefficient of determination): $R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \bar{y})^2}$
- Mean Absolute Error: $\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \left| y_i - \hat{y}_i \right|.$
- Median Absolute Error: $\text{MedAE}(y, \hat{y}) = \text{median}(\mid y_1 - \hat{y}_1 \mid, \ldots, \mid y_n - \hat{y}_n \mid).$
- Mean Squared Error: $\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$
- Mean Absolute Percentage Error: $\text{MAPE} = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}-1}} \frac{|y_i - \hat{y}_i|}{|y_i|}$
- Mean Squared Logarithmic Error: $\text{MSLE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (\log_e (1 + y_i) - \log_e (1 + \hat{y}_i))^2.$

Stationarity

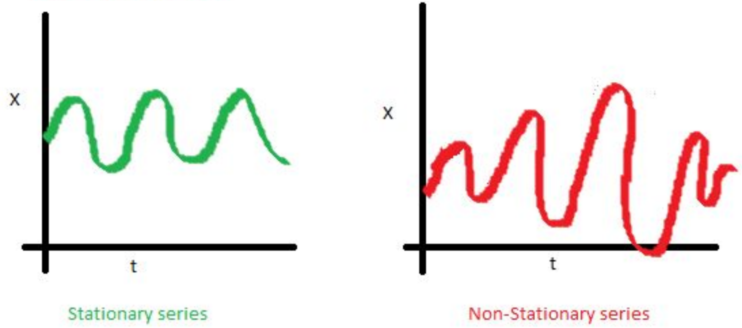
Stationary process does not change its statistical properties (mean and variance) over time.



a)



c)



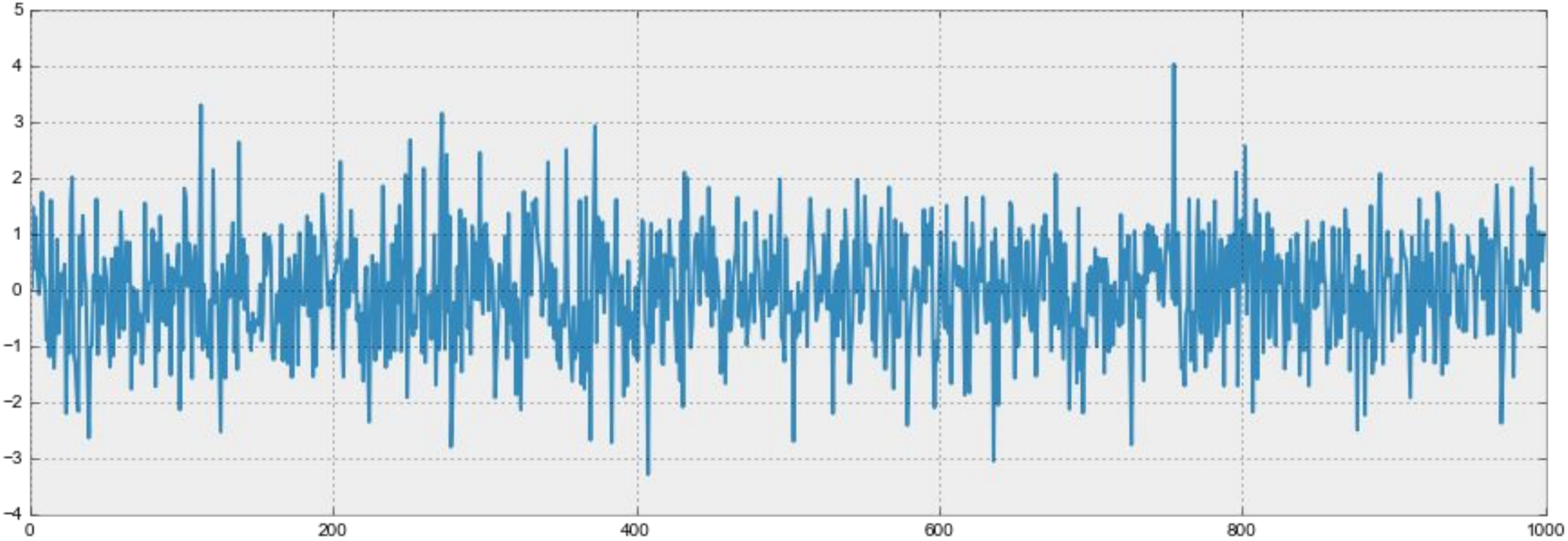
b)

a) Increasing mean
b) Variance changing over time
c) Covariance changes over time

Source: [post by Sean Abu]

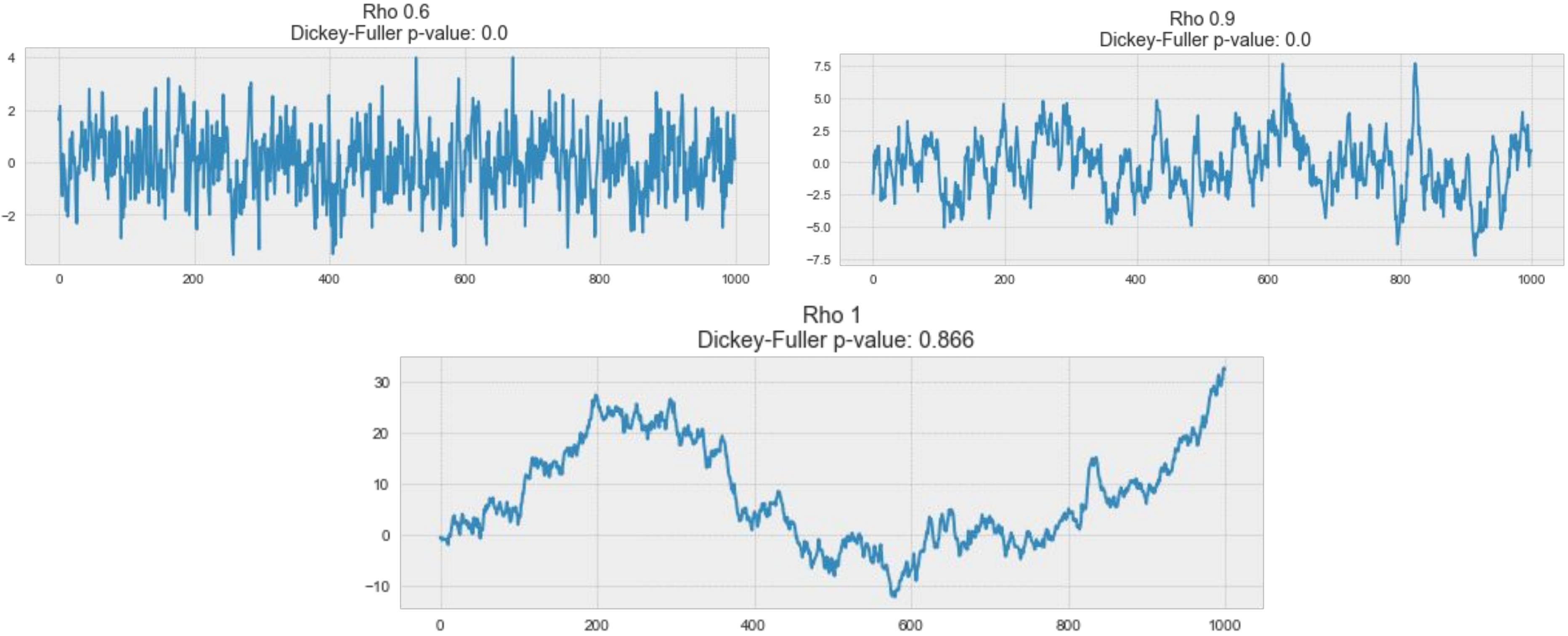
White noise

Process oscillates around 0 with deviation of 1.



White noise to random walk

Let's transform the white noise process: $y(t)=\rho*y(t-1)+e(t)$

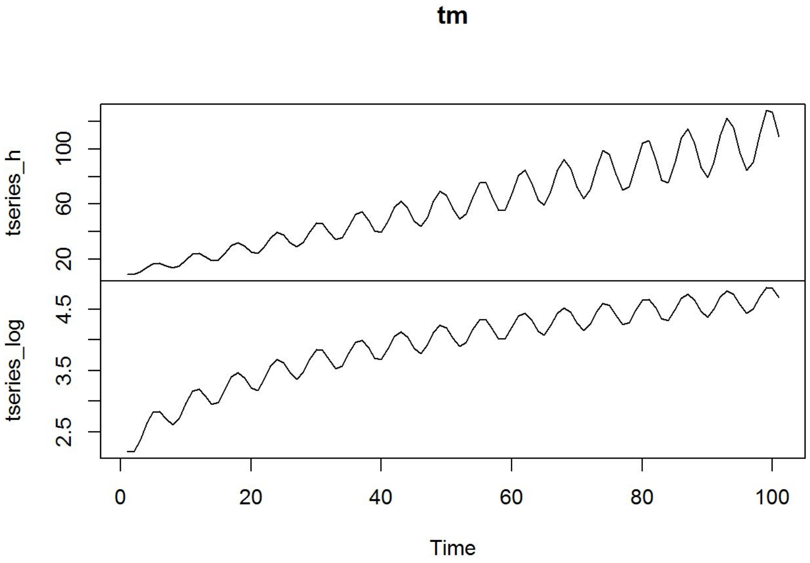


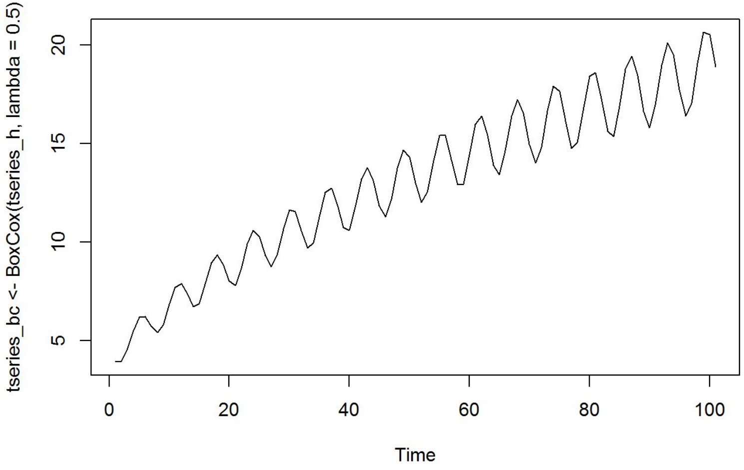
Augmented Dickey-Fuller test in statsmodels: link

Variance stabilization

Box-Cox transformation:

$$w_t = \begin{cases} \log y_t, & \text{if } \lambda = 0; \\ \frac{(y_t{}^{\lambda}-1)}{\lambda}, & \text{otherwise} \end{cases}$$





Source: [datascienceplus blog post]

Averaging and smoothing

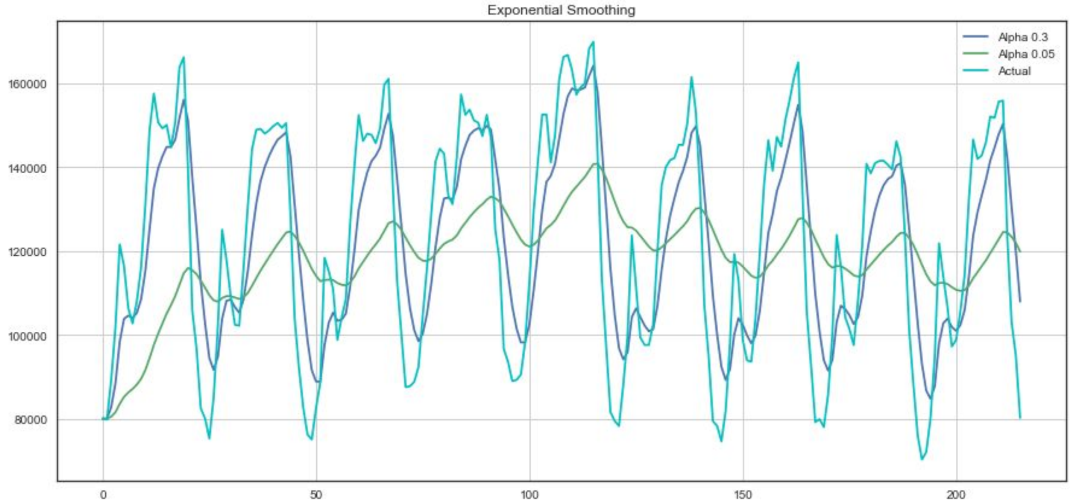
- Moving average: $$\hat{y}_t = \frac{1}{k}\sum_{n=0}^{k-1} y_{t-n}$$

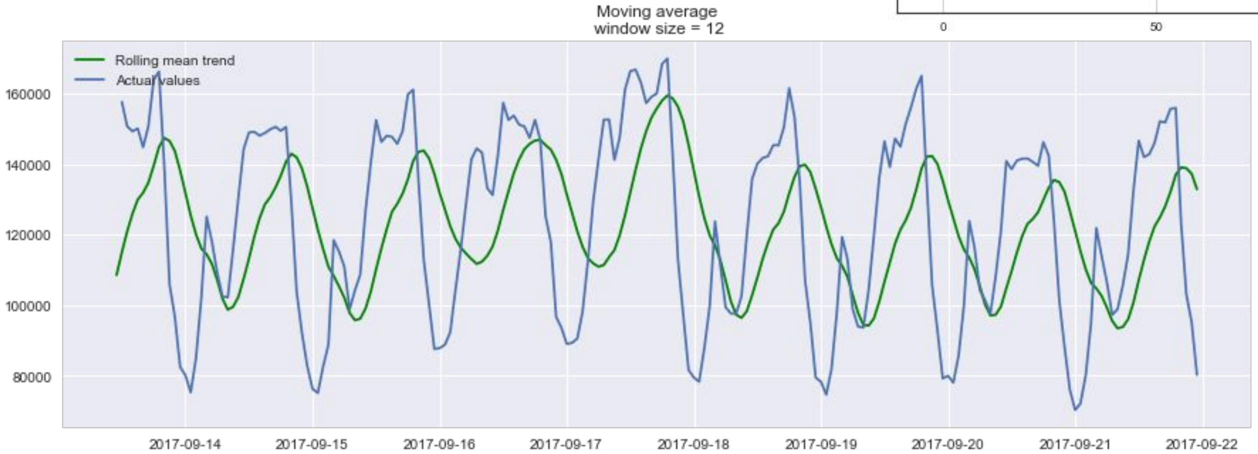
- Weighted moving average: $$\hat{y}_t = \sum_{n=1}^{k} \omega_n y_{t+1-n}$$

- Exponential smoothing: $$\hat{y}_t = \alpha \cdot y_t + (1-\alpha) \cdot \hat{y}_{t-1}$$

Averaging and smoothing

Caution: only one step is predicted





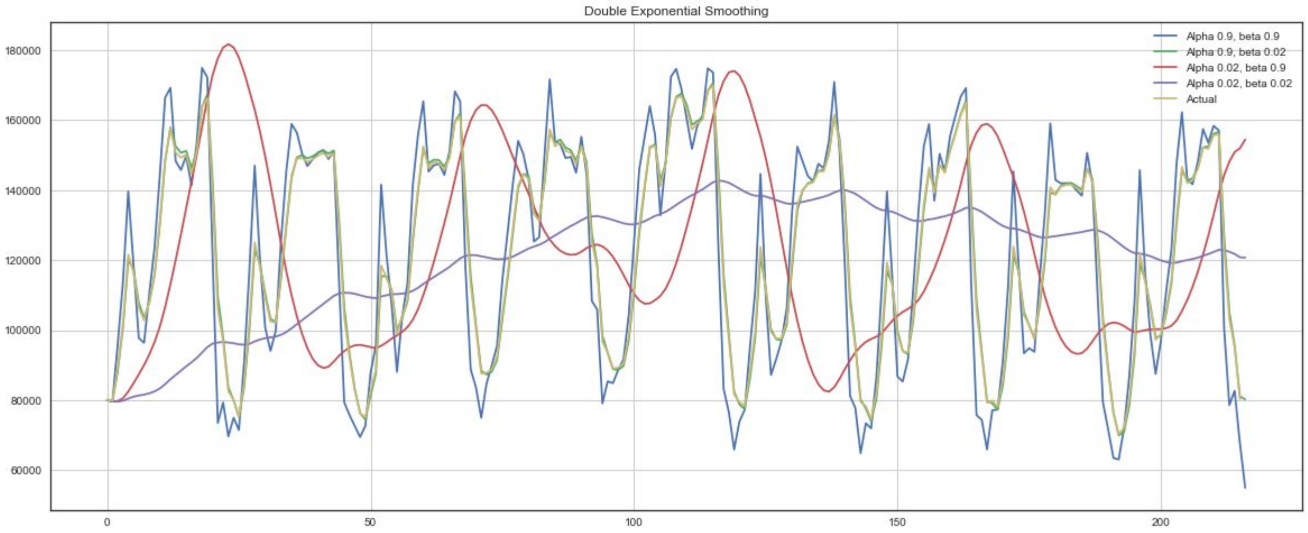
Double exponential smoothing

Two components:

intercept (or level) : $$\ell_x = \alpha y_x + (1-\alpha)(\ell_{x-1} + b_{x-1})$$

trend (or slope): $$b_x = \beta(\ell_x - \ell_{x-1}) + (1-\beta)b_{x-1}$$

$$\hat{y}_{x+1} = \ell_x + b_x$$



Now two steps can be predicted

Triple exponential smoothing (Holt-Winters)

Let’s add one more component: seasonality (s).

Seasonal component in the model should explain repeated variations around intercept and trend, and it will be described by the length of the season.

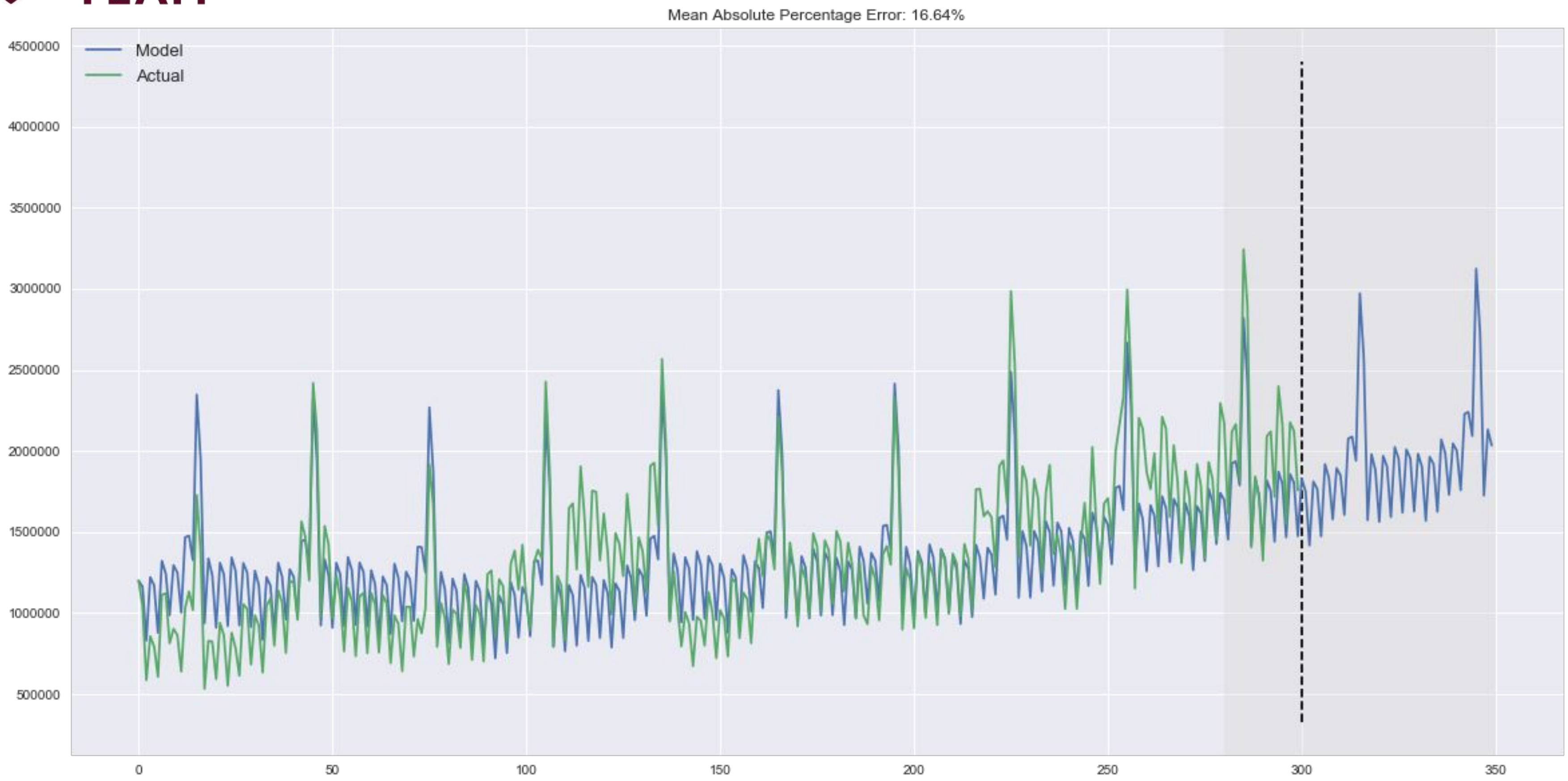
$$\ell_x = \alpha(y_x - s_{x-L}) + (1-\alpha)(\ell_{x-1} + b_{x-1})$$

$$b_x = \beta(\ell_x - \ell_{x-1}) + (1-\beta)b_{x-1}$$

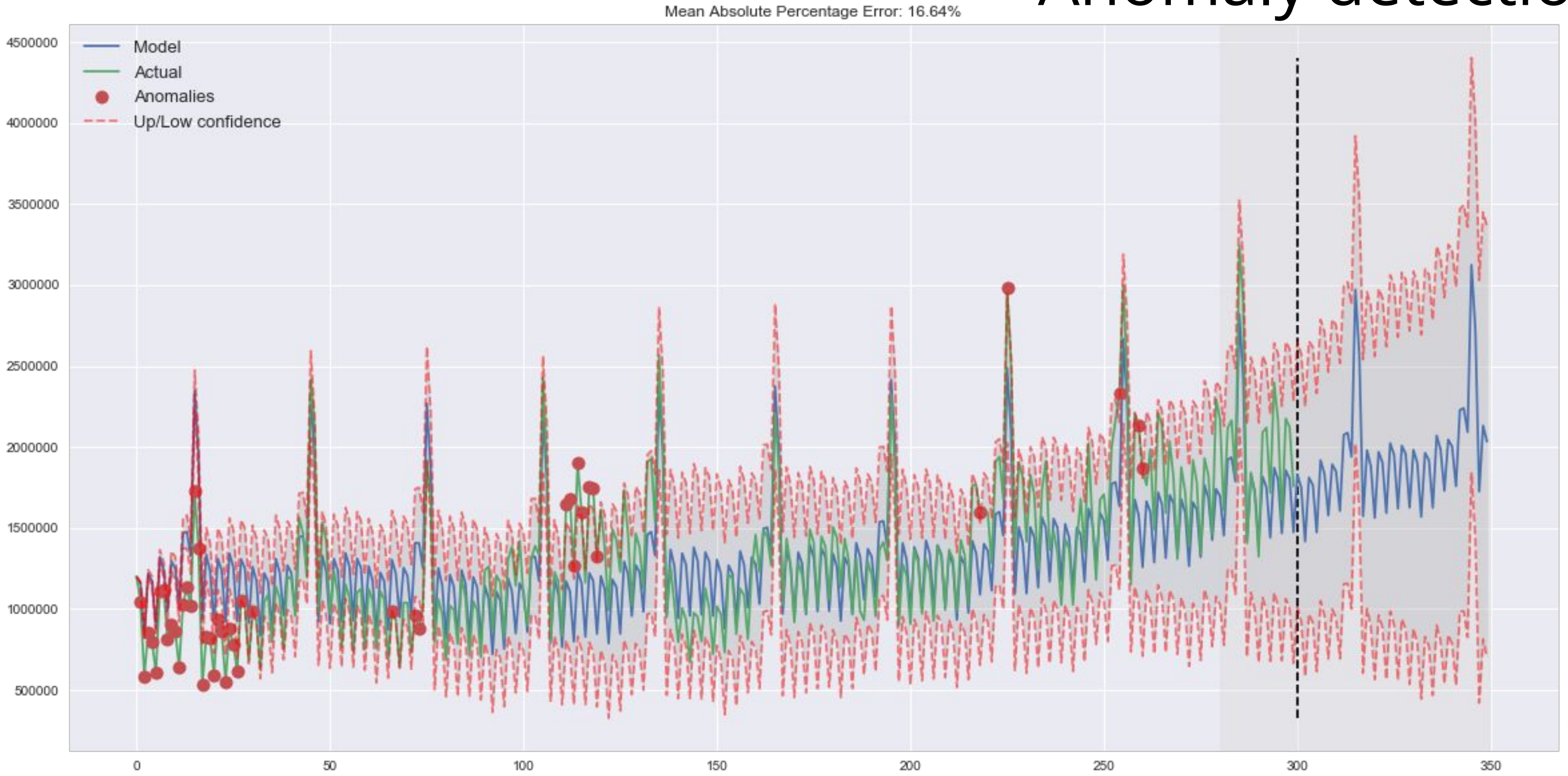
$$s_x = \gamma(y_x - \ell_x) + (1-\gamma)s_{x-L}$$

$$\hat{y}_{x+m} = \ell_x + mb_x + s_{x-L+1+(m-1)modL}$$

Triple exponential smoothing example



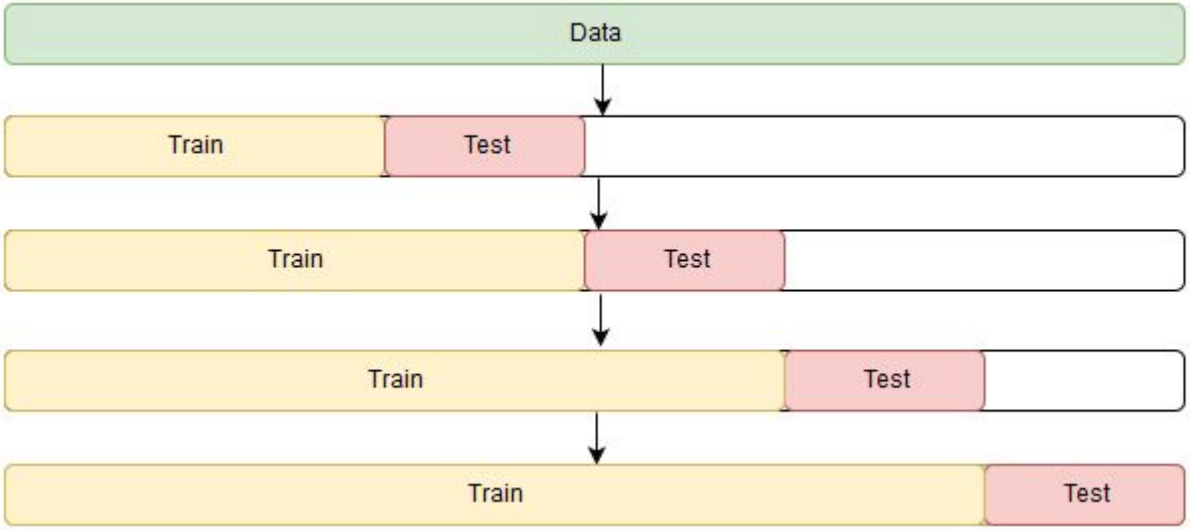
Triple exponential smoothing example: Anomaly detection



Actually, simple exponential smoothing allows to detect anomalies as well.

One more great approach from econometrics: ARIMA (link1, link2, link3 etc.)

Time Series Cross Validation



BIGDATA TEAM

Time to take a break

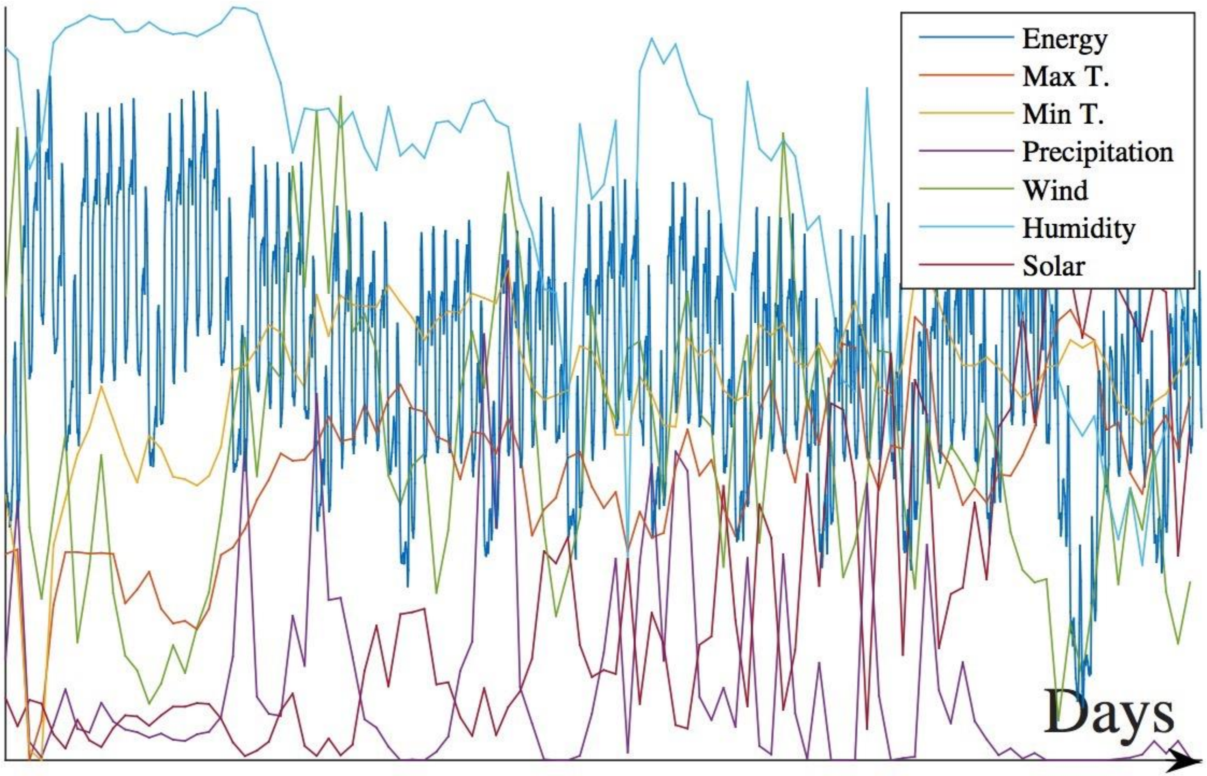
But first, feedback, please: http://bit.ly/ml4megafon_august18_lecture5_feedback

Feature generation

Different ways to generate features:

- Lagged Time Series Values
- Rolling statistics (max, min, median, mean, variance etc. in some window)
- Data-based features (information about holidays, siesta hours, special events etc)
- Exogenous variables
- Predictions of some extra models

Example TS with exogenous variables

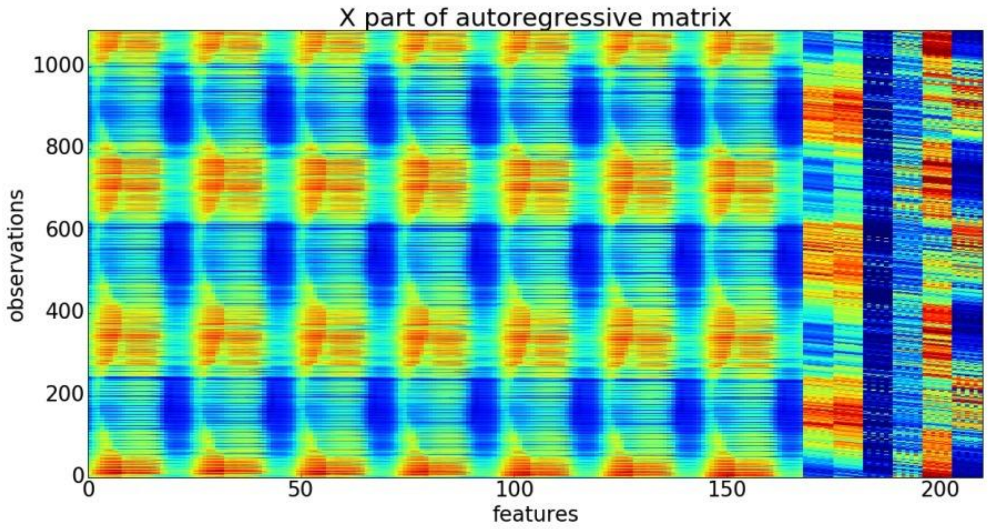


Wait, what about classification problem?

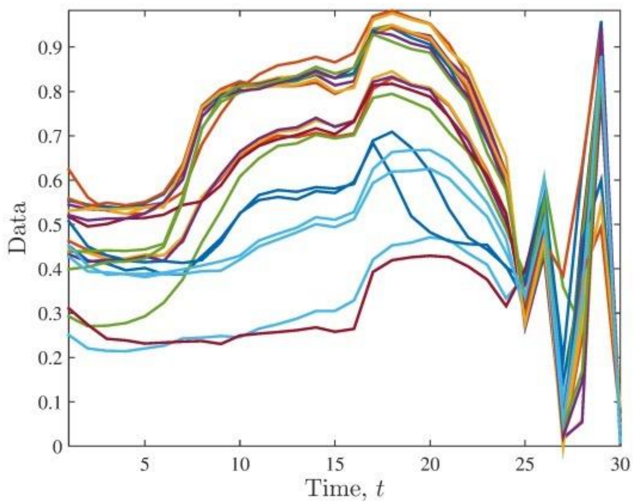
Wait, what about classification problem?

Easy: now we are just dealing with supervised learning problem!

Example Time Series: design matrix



a)



b)

a) Design matrix example
b) Target variables example

Supervised learning approaches

And here come all the great methods:

- Linear models
- Random Forest
- XGBoost (and alike)
- Neural Networks (RNN, CNN etc.)

To describe our data well, we need good feature space.

And feature engineering is an *art*.

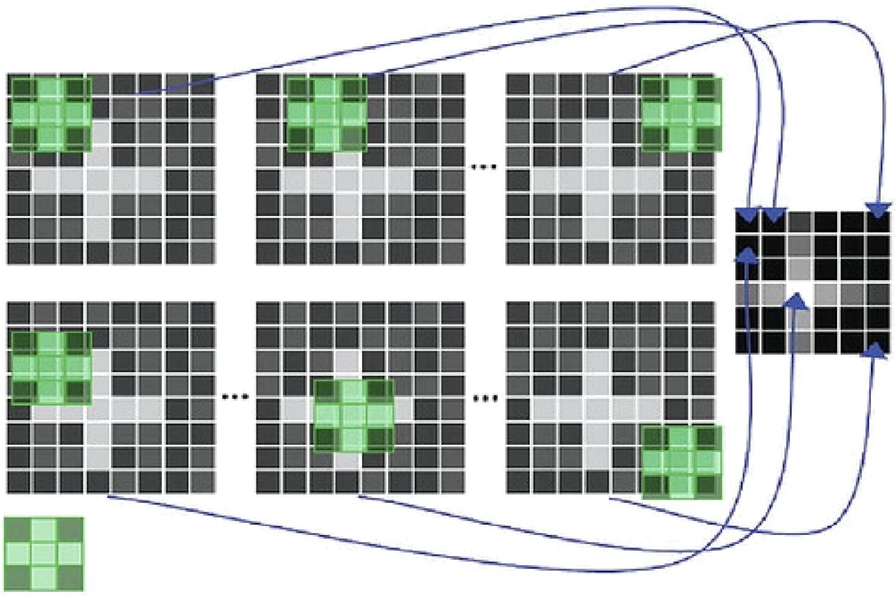
Possible way out:

1. Neural Networks

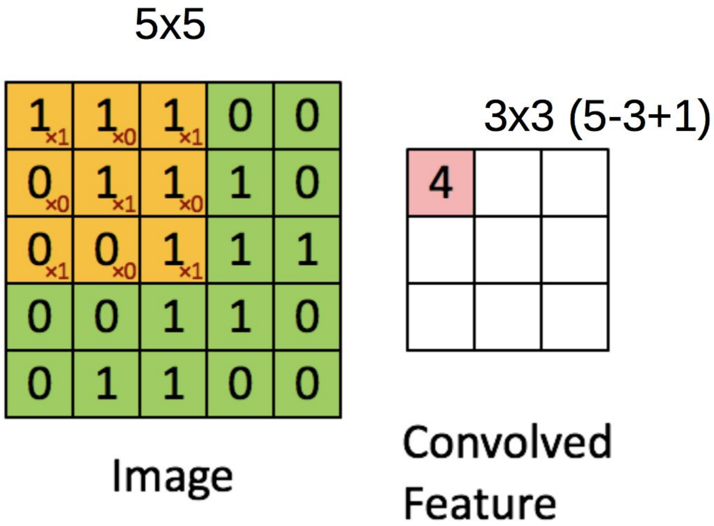
Recap: convolution layer

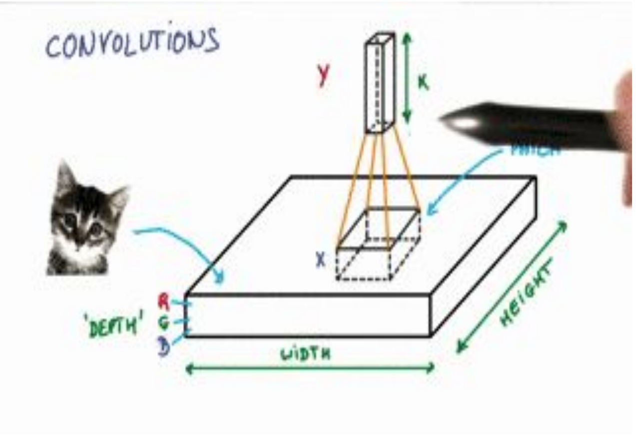
2d (and 3d) convolution: vital for Computer Vision problems.

1d convolution fits time series analysis.



Recap: convolution layer

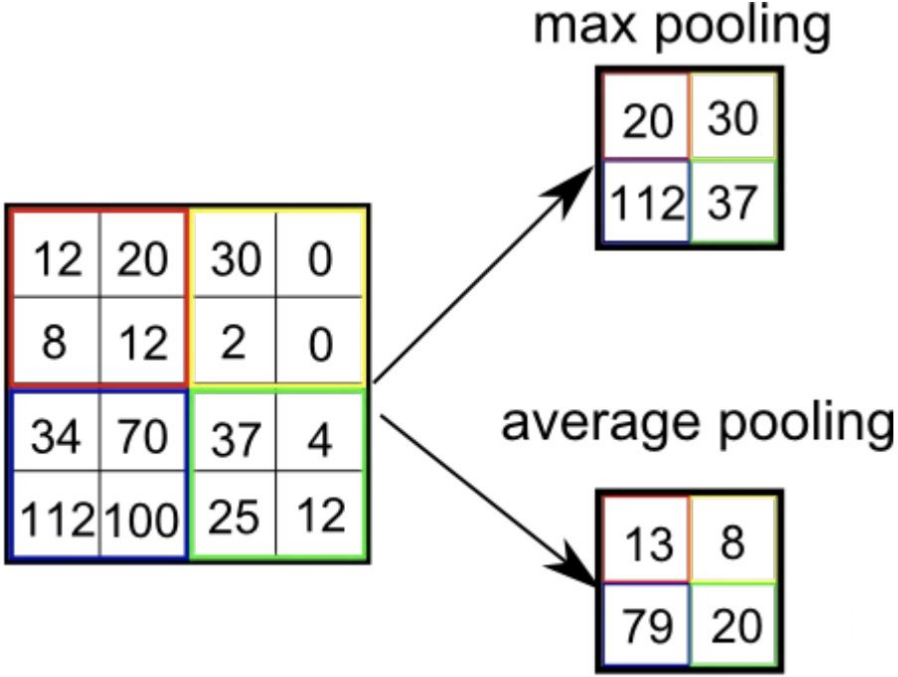




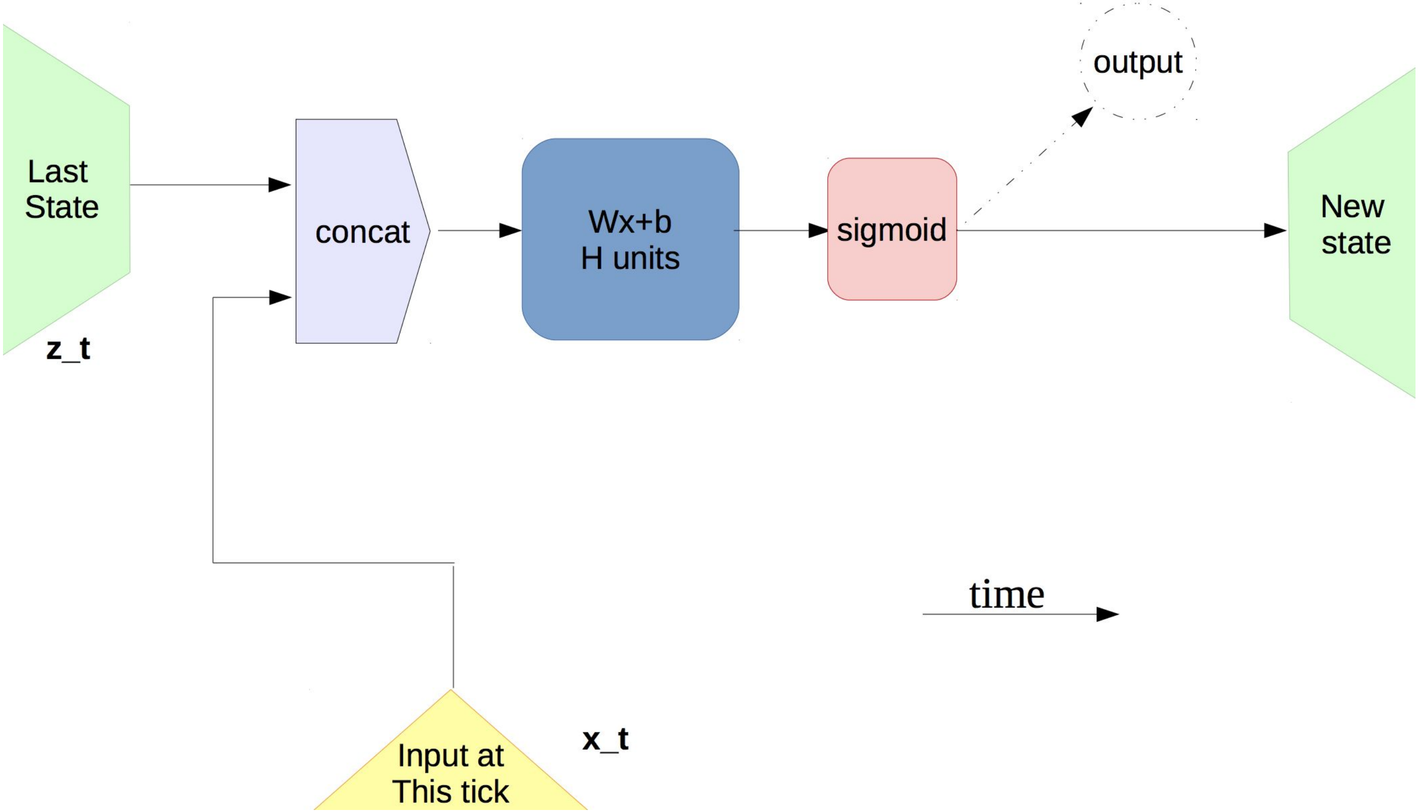
Intuition: how *cat-like* is this square?

Recap: pooling layer

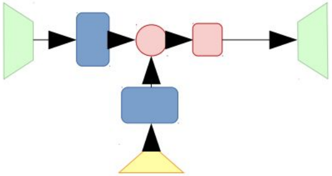
- Reduces layer size by a factc
- Makes NN less sensitive to small signal (image/ts) shifts
-
- Widely used:
  - max pooling
  - mean pooling



Recap : recurrent neural network

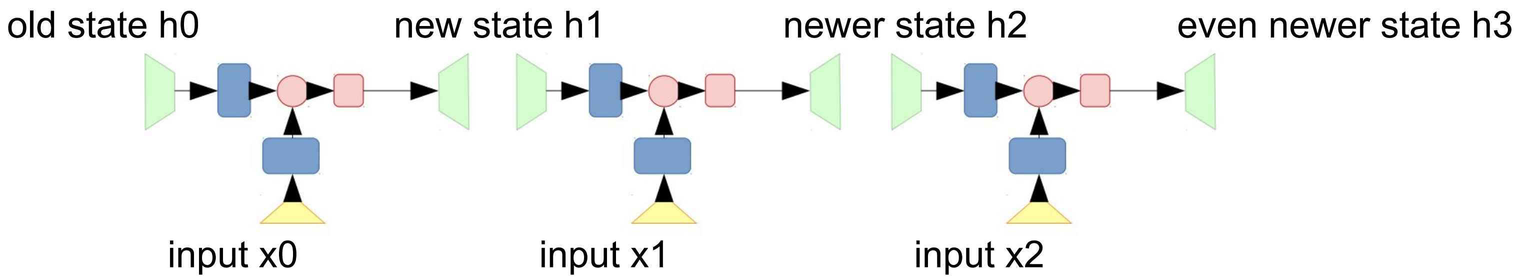


Recap : recurrent neural network

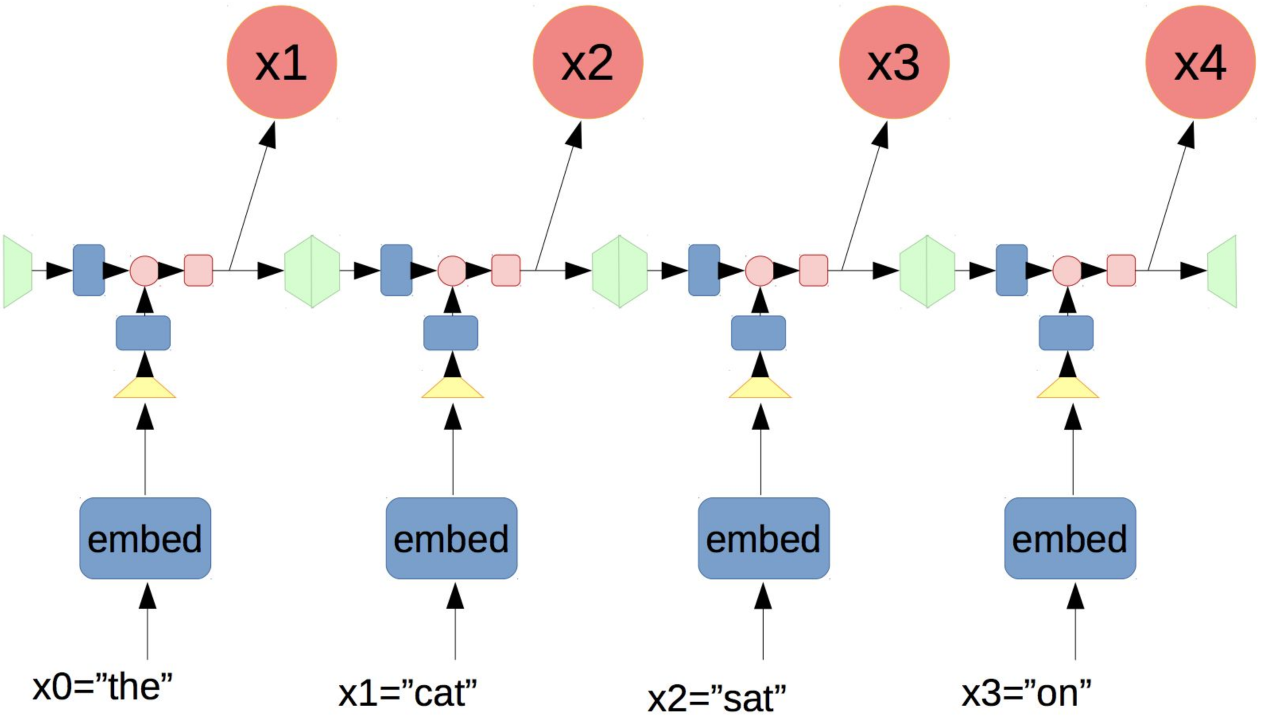


Recap : recurrent neural network

We use same weight matrices for all steps



Recap : recurrent neural network



Recap : recurrent neural network

Now with formulas

$$h_0 = \bar{0}$$

$$h_1 = \sigma\Big(\langle W_{\text{hid}}[h_0, x_0]\rangle + b\Big)$$

$$h_2 = \sigma\Big(\langle W_{\text{hid}}[h_1, x_1]\rangle + b\Big) = \sigma\Big(\langle W_{\text{hid}}[\sigma\Big(\langle W_{\text{hid}}[h_0, x_0]\rangle + b, x_1]\rangle + b\Big)$$

$$h_{i+1} = \sigma\Big(\langle W_{\text{hid}}[h_i, x_i]\rangle + b\Big)$$

$$P(x_{i+1}) = \text{softmax}\Big(\langle W_{\text{out}}, h_i\rangle + b_{\text{out}}\Big)$$

To describe our data well, we need good feature space.

And feature engineering is an *art*.

Possible way out:

1. Neural Networks
2. Some libraries that do this stuff for us (Facebook Prophet e.g.). But it's coming next

BIGDATA TEAM

That's all. Time to get some practice.

But first, feedback, please:

http://bit.ly/ml4megafon_august18_lecture6_feedback